APPENDIX A

C:\Documents and Settings\jhuggins\Local Settings\Temporary Internet Files\OLK4\Oper10/31/2001 6:00PM

```
//this is an example of intercepting an opengl call, and converting it into dual Direct3d8.
//This is one of the simplest examples possible.
//Some functions dont require much work at all.
//Other functions require extremly complex data conversion.
//this ClearDepth function, happens to be very similar to its d3d8 equavalent function
//  we know the val for depth is [0-1] which is same for input of d3d8's clear function.
//  thus no conversion of data required, just redirection.
//  If any conversion is required, it is done inside Opengl32.cpp.

//-------------------------------------------------------------------------------------------
//OPENGL32.CPP
//header for real function, written by SGI OpenGL
void (__stdcall* real_glClearDepth)(GLclampd depth);


//During init, we retrieve a pointer to the real opengl function
real_glClearDepth  = (void(__stdcall*)(GLclampd depth))GetProcAddress(DLLInst,"glClearDepth");



//inside our opengl32.dll wrapper, our pseudo function looks like this :
__declspec(dllexport) void __stdcall glClearDepth(GLclampd depth)
{
    if(convertTOd3d8)
    {//activly converting stream into d3d8dual
        //preform any necesary data conversion here.
        d3d_glClearDepth(depth);
    }
    else
    {//pass through, debug mode. normal OpenGL operation.
        real_glClearDepth(depth);
    }
}


//-------------------------------------------------------------------------------------------
//DUAL.CPP
//The opengl32.dll wrapper calls this function provided by our DualRendering System.
void d3d_glClearDepth(float depth)
{
    dual_glClearDepth(depth);
}

//the dual glClearDepth issues the commands to the 2 video cards.
void dual_glClearDepth(float depth)
{
    if(g_d3ddev1 != NULL)
    {
        g_d3ddev1->Clear(0,NULL,D3DCLEAR_ZBUFFER,D3DCOLOR_XRGB(0x00,0x00,0x00),depth,0);
    }
    if(g_d3ddev2 != NULL)
    {
        g_d3ddev2->Clear(0,NULL,D3DCLEAR_ZBUFFER,D3DCOLOR_XRGB(0x00,0x00,0x00),depth,0);
    }
}
```

C:\Documents and Settings\jhuggins\Local Settings\Temporary Internet Files\OLK4\Glid10/31/2001 5:00PM

```
//GLIDE EXAMPLE of dual rendering
//Glide openly allows access to 2 cards by calling grSstSelect(0), or 1
//Glide also doesnt have to worry about "exclusive mode" which only allows 1 full screen DrectX wind
ow.
//  So no special code for window creation is necessary.
//Due to differences in the API's, the data at this point has already been transformed from 3D into
2D data.
//As a result, less accurate method of creating stereo image is allpied.
//  This stereo method moves the geometry(in 2D), rather than the correct method of moving the camer
a.
//Assembly was used to bypass the C/C++ const barrier. In assembly, it is not "read only"
//  const means "read only" "you cant modify it legally"
//  In assembly language, the "read only" lock is not checked.
//  This allows us to move the const geometry.
//  The assembly simply adds, or subtracts an offset, based on the geometrys distance from camera.

FX_ENTRY void FX_CALL PgrDrawTriangle(const GrVertex *a,
                                      const GrVertex *b,
                                      const GrVertex *c, float& angle, float& limit)
{
    float dista = (a ->oow) * angle;
        if ( abs((int)dista) >= abs((int)limit))
            dista = limit;
    float distb = (b->oow) * angle;
        if ( abs((int)distb) >= abs((int)limit))
            distb = limit;
    float distc = (c->oow) * angle;
        if (abs((int)distc) >= abs((int)limit))
            distc = limit;

    float temporaire = 0.0f;
        //On commence par soustraire le decalage

    __asm
    {
        //Premier point
        pushad
        push ds
        mov esi, a
        mov eax, [esi]
        mov temporaire, eax
        fld temporaire
        fsub dista
        fstp temporaire
        mov eax, temporaire
        mov [esi],eax
        //Deuxieme point
        mov esi,b
        mov eax,[esi]
        mov temporaire,eax
        fld temporaire
        fsub distb
        fstp temporaire
        mov eax,temporaire
        mov [esi],eax
        //Troisieme point
        mov esi,c
        mov eax,[esi]
        mov temporaire,eax
        fld temporaire
        fsub distc
        fstp temporaire
        mov eax,temporaire
        mov [esi],eax
        pop ds
        popad
    }

    REAL_grDrawTriangle(a, b, c);
```

```
    dista = 2 * dista;
    distb = 2 * distb;
    distc = 2 * distc;

    __asm
    {
        //Premier point
        pushad
        push ds
        mov esi, a
        mov eax, [esi]
        mov temporaire, eax
        fld temporaire
        fadd dista
        fstp temporaire
        mov eax, temporaire
        mov [esi],eax
        //Deuxieme point
        mov esi,b
        mov eax,[esi]
        mov temporaire,eax
        fld temporaire
        fadd distb
        fstp temporaire
        mov eax,temporaire
        mov [esi],eax
        //Troisieme point
        mov esi,c
        mov eax,[esi]
        mov temporaire,eax
        fld temporaire
        fadd distc
        fstp temporaire
        mov eax,temporaire
        mov [esi],eax
        pop ds
        popad
    }
    REAL_grSstSelect(1);
    REAL_grDrawTriangle(a, b, c);

    //Restoration
    dista =  dista / 2;
    distb =  distb / 2;
    distc =  distc / 2;

        __asm
    {
        //Premier point
        pushad
        push ds
        mov esi, a
        mov eax, [esi]
        mov temporaire, eax
        fld temporaire
        fsub dista
        fstp temporaire
        mov eax, temporaire
        mov [esi],eax
        //Deuxieme point
        mov esi,b
        mov eax,[esi]
        mov temporaire,eax
        fld temporaire
        fsub distb
        fstp temporaire
        mov eax,temporaire
        mov [esi],eax
        //Troisieme point
        mov esi,c
        mov eax,[esi]
        mov temporaire,eax
```

```
        fld temporaire
        fsub distc
        fstp temporaire
        mov eax,temporaire
        mov [esi],eax
        pop ds
        popad
    }

REAL_gr$stSelect(0);

}
```

10011027.110201

```
//Windows specific code for creation of 2 full screen windows
//Function is called twice, once for each display. 2 displays for 2 eyes.

bool WindowCreate(  int Id,
                    HINSTANCE hInstance,
                    char* pWindowName,
                    char* pClassName,
                    HWND& hwnd,
                    HWND& parenthwnd)
{
    WNDCLASS wc;

    wc.style            = 0;
    if(Id==0)
    {
        wc.lpfnWndProc      = (WNDPROC) WndProc1;
    }
    else if(Id==1)
    {
        wc.lpfnWndProc      = (WNDPROC) WndProc1;
    }
    else
    {
        assert(0);
    }
    wc.cbClsExtra       = 0;
    wc.cbWndExtra       = 0;
    wc.hInstance        = hInstance;
    wc.hIcon            = NULL;
    wc.hCursor          = (HCURSOR) NULL;
    wc.hbrBackground    = (HBRUSH)COLOR_INACTIVECAPTION;
    wc.lpszMenuName     = NULL;
    wc.lpszClassName    = pClassName;

    if (!RegisterClass(&wc))
    {
        sprintf(pDebugText,"RegisterClass(&wc) FAILED\n");
        OutDebugErrorMsg();
        return false;
    }

    int thisone = 0;
    //this part is critical for Atlantis. Allows 2 FULL SCREEN, Hardware accelerated windows
    //the poorly documented WS POPUP|WS VISIBLE flags make a
    //      window without borders. ie windowed, but FULL SCREEN
    //2 "real" FULLSCREENS is impossible, because first "real" FULLSCREEN sets exclusive mode.

    hwnd = CreateWindow(pClassName,
                        pWindowName,
                        WS_POPUP|WS_VISIBLE ,
                        CW_USEDEFAULT,
                        CW_USEDEFAULT,
                        ScreenWidth,
                        ScreenHeight,
                        parenthwnd,
                        NULL,
                        hInstance,
                        NULL);

    // If the main window cannot be created, terminate
    // the application.
    if (hwnd == 0)
    {
        sprintf(pDebugText,"hwnd==NULL : FAILED\n");
        OutDebugErrorMsg();
        return false;
    }

    if(Id==0)
    {
        //position first window at 0,0 on monitor 1 assumed to be at 640x480
```

```
        SetWindowPos(hwnd,HWND_TOPMOST,0,0,ScreenWidth,ScreenHeight,SWP_SHOWWINDOW );
    }
    else if(Id==1)
    {
        //position second window at 0,0 on monitor 2 assumed to be at 640x480
        SetWindowPos(hwnd,HWND_TOPMOST,ACTUALScreenWidth,0,ScreenWidth,ScreenHeight,SWP_SHOWWINDOW )
;
    }
    return true;
}

//D3D8 creation of 2 devices
//debug #defines. allows for programmer to debug system using 1, or 2, or both devices simultaneousl
y.
//for release, both are defined.
//  ACCELERATOR_1_AVAILABLE
//  ACCELERATOR_2_AVAILABLE
int InitializeHardware(HINSTANCE hInstance)
{
    WNDCLASS wc1;
    WNDCLASS wc2;

    static char *CLASS_NAME1  = "CLASS1";
    static char *CLASS_NAME2  = "CLASS2";
    static char *WINDOW_NAME1 = "Window 1";
    static char *WINDOW_NAME2 = "Window 2";

    DiskFile=fopen("c:\\backup\\DualTest.TXT","w");
    fprintf(DiskFile,"Atlantis Cyberspace\n");
    fclose(DiskFile);

    sprintf(pDebugText,"~InitializeHardware~\n");
    OutDebugErrorMsg();
    //_____

    HWND DesktopWindow = GetDesktopWindow();
    WindowCreate(0,hInstance,WINDOW_NAME1,CLASS_NAME1,g_hwnd1,DesktopWindow);

    #ifdef ACCELERATOR_2_AVAILABLE
    WindowCreate(1,hInstance,WINDOW_NAME2,CLASS_NAME2,g_hwnd2,g_hwnd1);
    #endif//ACCELERATOR_2_AVAILABLE

    //_____


    #ifdef ACCELERATOR_1_AVAILABLE
    pEnum = Direct3DCreate8(D3D_SDK_VERSION);
    if (pEnum == NULL)
    {
        sprintf(pDebugText,"Direct3DCreate8 Device 1 : FAILED\n");
        OutDebugErrorMsg();
        return -1;
    }
    #endif//ACCELERATOR_1_AVAILABLE


    #ifdef ACCELERATOR_2_AVAILABLE
    pEnum2 = Direct3DCreate8(D3D_SDK_VERSION);
    if (pEnum2 == NULL)
    {
        sprintf(pDebugText,"Direct3DCreate8 Device 2 : FAILED\n");
        OutDebugErrorMsg();
        return -1;
    }
    #endif//ACCELERATOR_2_AVAILABLE

    //_____

    #ifdef ACCELERATOR_1_AVAILABLE
    DeviceCreate(g_hwnd1,pEnum,g_d3ddev1,D3DADAPTER_DEFAULT);
    #endif//ACCELERATOR_1_AVAILABLE
```

```
    #ifdef ACCELERATOR_2_AVAILABLE
    DeviceCreate(g_hwnd2,pEnum2,g_d3ddev2,1);
    #endif//ACCELERATOR_2_AVAILABLE

    //-----------------------------------------------------------------------

    #ifdef ACCELERATOR_1_AVAILABLE
    ShowWindow(g_hwnd1, SW_SHOWDEFAULT);
    UpdateWindow(g_hwnd1);
    #endif//ACCELERATOR_1_AVAILABLE

    #ifdef ACCELERATOR_2_AVAILABLE
    ShowWindow(g_hwnd2, SW_SHOWDEFAULT);
    UpdateWindow(g_hwnd2);
    #endif//ACCELERATOR_2_AVAILABLE

    //-----------------------------------------------------------------------
    if(g_d3ddev1)
    {
        g_d3ddev1->SetRenderState(D3DRS_LIGHTING,FALSE);
        g_d3ddev1->SetRenderState(D3DRS_ALPHABLENDENABLE,FALSE);
        g_d3ddev1->SetRenderState(D3DRS_FILLMODE,D3DFILL_SOLID);

        g_d3ddev1->SetRenderState(D3DRS_CLIPPING,TRUE);

        g_d3ddev1->SetRenderState(D3DRS_ZENABLE,FALSE);
        g_d3ddev1->SetRenderState(D3DRS_ZWRITEENABLE,FALSE);

        g_d3ddev1->SetTextureStageState(0,D3DTSS_MINFILTER,D3DTEXF_LINEAR);
        g_d3ddev1->SetTextureStageState(0,D3DTSS_MAGFILTER,D3DTEXF_LINEAR);
        g_d3ddev1->SetTextureStageState(0, D3DTSS_MIPFILTER, D3DTEXF_POINT);
    }
    if(g_d3ddev2)
    {
        g_d3ddev2->SetRenderState(D3DRS_LIGHTING,FALSE);
        g_d3ddev2->SetRenderState(D3DRS_ALPHABLENDENABLE,FALSE);
        g_d3ddev2->SetRenderState(D3DRS_FILLMODE,D3DFILL_SOLID);

        g_d3ddev2->SetRenderState(D3DRS_CLIPPING,TRUE);

        g_d3ddev2->SetRenderState(D3DRS_ZENABLE,FALSE);
        g_d3ddev2->SetRenderState(D3DRS_ZWRITEENABLE,FALSE);

        g_d3ddev2->SetTextureStageState(0,D3DTSS_MINFILTER,D3DTEXF_LINEAR);
        g_d3ddev2->SetTextureStageState(0,D3DTSS_MAGFILTER,D3DTEXF_LINEAR);
        g_d3ddev1->SetTextureStageState(0,D3DTSS_MIPFILTER,D3DTEXF_POINT);
    }
    InitializeTextureManager();
    dual_RestoreVertexBuffers();
    ResetBindTextureOrderList();
    d3d_InitMatrixStack(&g_ModelViewStack );
    d3d_InitMatrixStack(&g_ProjectionStack);

    g_Viewport.X      = 0;
    g_Viewport.Y      = 0;
    g_Viewport.Width  = 640;
    g_Viewport.Height = 480;
    g_Viewport.MinZ   = 0.0;
    g_Viewport.MaxZ   = 1.0;
    return 0;
}
```

```
//This function is one of many that handle the rendering.
// Other functions similar to this one are : RenderTriangle, RenderQuad, RenderTriangleStrip... etc.
```

C:\Documents and Settings\jhuggins\Local Settings\Temporary Internet Files\OLK4\d3d810/31/2001 5:01 PM

```
//the global variables g_d3ddev1, and g_d3ddev2 are pointers to IDirect3DDevice8.
//a IDirect3DDevice8 can be thought of as the last software interface to the video card.
//most commands are issued twice.
//After a g_d3ddev command is issued, it immediatly returns, so that execution can continue.
//  This allows for concurency. The first card starts rendering, ans the second card is recieving da
ta.
//  At some point, they are both rendering, and Intel CPU is free to continue doing other things, wh
ile video cards render to their own memory.
void RenderTriangleFan(MYVERTEX2* pVertices,long num_verts)
{
    if(g_d3ddev1 != NULL)
    {
        assert(state_d3ddev1==1);
    }
    if(g_d3ddev2 != NULL)
    {
        assert(state_d3ddev2==1);
    }

    HRESULT Error = S_OK;
    HRESULT hr = S_OK;
    MYVERTEX2 Quad[1024];
    long i;

    ////////////////////////////////////////////////////////////////////////////////////////////
    if(g_d3ddev1 != NULL)
    {
        FrameCounter++;

        g_d3ddev1->SetVertexShader(D3DFVF_D3DVERTEX);
#ifdef USE_SET_TEXTURE
        g_d3ddev1->SetTexture( 0, p_gl_TEXTURE[c_glBindTexture].pD3DTexture0);
#endif
        if(max_num_verts<num_verts)
        {
            max_num_verts=num_verts;
        }

        if(bWriteToForground)
        {
            g_d3ddev1->SetRenderState(D3DRS_ZENABLE,TRUE);
            g_d3ddev1->SetRenderState(D3DRS_ZWRITEENABLE,FALSE);
        }
        else if(bWriteToBackground)
        {
            g_d3ddev1->SetRenderState(D3DRS_ZENABLE,TRUE);
            g_d3ddev1->SetRenderState(D3DRS_ZWRITEENABLE,FALSE);
        }
        else
        {
            g_d3ddev1->SetRenderState(D3DRS_ZENABLE,     bZBufferRead );
            g_d3ddev1->SetRenderState(D3DRS_ZWRITEENABLE,bZBufferWrite);
        }
#ifdef RENDER_POLYGONS
        hr = g_d3ddev1->DrawPrimitiveUP(D3DPT_TRIANGLEFAN,num_verts-2,pVertices,sizeof(MYVERTEX2));
        total_num_verts += num_verts;
        total_num_tris  += num_verts-2;
#endif//RENDER_POLYGONS
        if(FAILED(hr))
        {
            sprintf(pDebugText,"g_d3ddev1->DrawPrimitiveUP : FAILED\n");
            OutDebugErrorMsg();
            GetError(hr);
            OutDebugErrorMsg();
        }
    }
    ////////////////////////////////////////////////////////////////////////////////////////////
    if(g_d3ddev2 != NULL)
    {
        FrameCounter++;

        g_d3ddev2->SetVertexShader(D3DFVF_D3DVERTEX);
```

C:\Documents and Settings\jhuggins\Local Settings\Temporary Internet Files\OLK4\d3d10/31/2001 5:01 PM

```
#ifdef USE_SET_TEXTURE
        g_d3ddev2->SetTexture( 0, p_gl_TEXTURE[c_glBindTexture].pD3DTexture1);
#endif

        if(bWriteToForground)
        {
            g_d3ddev2->SetRenderState(D3DRS_ZENABLE,TRUE);
            g_d3ddev2->SetRenderState(D3DRS_ZWRITEENABLE,FALSE);
        }
        else if(bWriteToBackground)
        {
            g_d3ddev2->SetRenderState(D3DRS_ZENABLE,TRUE);
            g_d3ddev2->SetRenderState(D3DRS_ZWRITEENABLE,FALSE);
        }
        else
        {
            g_d3ddev2->SetRenderState(D3DRS_ZENABLE,     bZBufferRead );
            g_d3ddev2->SetRenderState(D3DRS_ZWRITEENABLE,bZBufferWrite);
        }

#ifdef RENDER_POLYGONS
        hr = g_d3ddev2->DrawPrimitiveUP(D3DPT_TRIANGLEFAN,num_verts-2,pVertices,sizeof(MYVERTEX2));
        total_num_verts += num_verts;
        total_num_tris  += num_verts-2;
#endif//RENDER_POLYGONS
        if(FAILED(hr))
        {
            sprintf(pDebugText,"g_d3ddev2->DrawPrimitiveUP : FAILED\n");
            OutDebugErrorMsg();
            GetError(hr);
            OutDebugErrorMsg();
        }
    }
    ////////////////////////////////////////////////////////////////////////////////////////////////////
    dual_SetZBias(0);
}
```

United States Patent & Trademark Office

Office of Initial Patent Examination -- Scanning Division

Application deficiencies found during scanning:

☐ Page(s)__________ of ________________________ were not present for scanning. (Document title)

☐ Page(s)__________ of ________________________ were not present for scanning. (Document title)

there are only 2 Pages of declaration.

☑ ***Scanned copy is best available.*** Some drawings are dark.